Christian Lipski,
Georgia Albuquerque,
Timo Stich,
Marcus Magnor

Spacetime Tetrahedra: Image-Based Viewpoint Navigation through Space and Time

Braunschweig : Computer Graphics Lab, 2008

Technical Report / Computer Graphics Lab, TU Braunschweig ; 2008-12-9

Veröffentlicht: 05.12.2008

http://www.digibib.tu-bs.de/?docid=00023968

Christian Lipski
lipski@cg.tu-bs.de
Computer Graphics Lab, TU Braunschweig

Georgia Albuquerque
georgia@cg.tu-bs.de
Computer Graphics Lab, TU Braunschweig

Timo Stich
stich@cg.tu-bs.de
Computer Graphics Lab, TU Braunschweig

Marcus Magnor
magnor@cg.tu-bs.de
Computer Graphics Lab, TU Braunschweig

Spacetime Tetrahedra: Image-Based Viewpoint Navigation through Space and Time

Technical Report 2008-12-9

December 3, 2008

Computer Graphics Lab, TU Braunschweig

Abstract

We present a purely image-based rendering system to viewpoint-navigate through space and time of arbitrary dynamic scenes. Unlike previous methods, our approach does not rely on synchronized and calibrated multi-video footage as input. Instead of estimating scene depth or reconstructing 3D geometry, our approach is based on dense image correspondences, treating view interpolation equally in space and time. In a nutshell, we tetrahedrally partition the volume spanned by camera directions and time, determine the warp field along each tetrahedral edge, and warp-blend-interpolate any viewpoint inside a tetrahedron from the four video frames representing its vertices. Besides fast and easy acquisition to make outdoor recordings feasible, our space-time symmetric approach allows for smooth interpolation of view perspective and time, i.e., for simultaneous free-viewpoint and slow motion rendering.

Contents

Chapter 1

Introduction

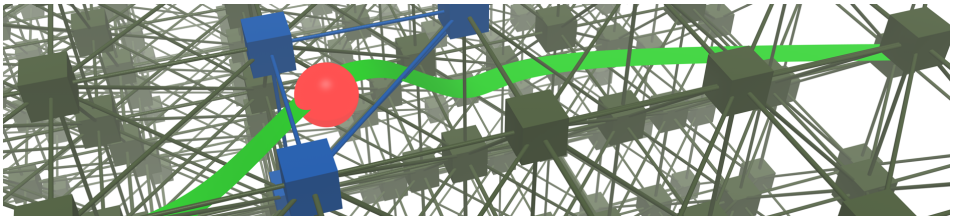

Figure 1.1: Spacetime tetrahedra: time and camera directions span our 3D navigation space. Each cube represents one video frame. The navigation space is partitioned into tetrahedra with video frames as vertices. Each tetrahedral edge denotes the correspondence field between two video frames. During interactive view navigation, the four video frames of the corresponding tetrahedron are warped and composited.

View interpolation is at the heart of any free-viewpoint navigation system [ZKU+04, CTMS03, VBK05]. Given a handful of simultaneously recorded video streams, the objective is to render photo-realistic views of some dynamic scene from any vantage point (within a specified viewpoint range). In order to be able to estimate depth or reconstruct 3D geometry, existing systems require synchronized video recordings and calibrated camera setups. This dependence on synchronized and calibrated multi-video footage, however, limits practical applicability: time-consuming and scene invading calibration procedures as well as synchronization wires between cameras make recordings in uncontrolled environments, e.g., of sports events or outdoors, very tedious.To overcome these drawbacks, we present a novel approach to free-viewpoint navigation that accepts unsynchronized, uncalibrated video streams as the only input.

The fundamental challenge with unsynchronized multi-video recordings is that it is unsuitable for depth or geometry analysis because image correspondences do not obey the epipolar constraint anymore, regardless of

whether cameras are calibrated or not. Instead, dense correspondences between images from different cameras must be established as 2D vector fields, e.g., via optical flow estimation.On the other hand, 2D correspondence fields are very versatile: they can be established, at least theoretically, for arbitrarily complex and unyielding scenes, and they can be estimated between images taken from different viewpoints as well as at different moments in time. This symmetry suggests to treat view interpolation in space (i.e., between camera positions) and time equally, based purely on dense image correspondence fields. In this manner, each video frame can be thought of as one sampling point in a 3-dimensional navigation space that is spanned by camera directions and time[1], Fig.1.1. All recorded multi-video frames taken together form a cloud of points whose outer hull defines the range of possible viewpoints that can be sensibly interpolated. We partition this point cloud into tetrahedra. To render the view corresponding to some arbitrary point in navigation space, the tetrahedron that contains the point is determined, and the view is interpolated from the four video frames representing the vertices of the tetrahedron. Besides smooth view interpolation between different cameras' perspectives, this approach simultaneously features smooth interpolation in time, i.e., we can render the dynamic scene from any perspective and at any moment in time. As contribution, our paper presents a system to viewpoint-navigate around arbitrary dynamic, real-world scenes from a handful of easy-to-acquire unsynchronized and uncalibrated video streams, at any play-back speed (e.g., slow motion).

Our system is divided into an offline pre-processing part and an online, real-time rendering algorithm for interactive navigation, Fig. 1.2. After highlighting relevant previous work in Sect. 2, we describe unsynchronized multi-video acquisition in Sect. 3. The paper's technical contributions are covered in Sects. 4 and 5 where we give a detailed account of all necessary pre-processing steps and the real-time rendering algorithm for interactive navigation. Results for various complex, real-world scenes are discussed in Sect. 6 before we conclude with Sect. 7.

[1] by camera direction, we refer to the direction of each camera's center of projection as seen from the center of the scene

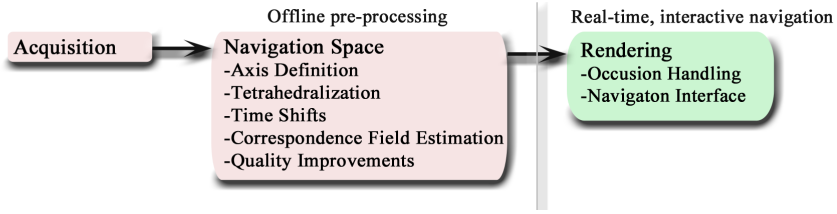

Figure 1.2: Processing pipeline: conventional consumer-grade video cameras are sufficient to capture arbitrary dynamic scenes as input. The unsynchronized, uncalibrated multi-video data is processed in an offline step, enabling interactive view navigation at real-time frame rates.

Chapter 2

Related Work

Image interpolation has received a lot of attention in computer graphics as well as computer vision. To interpolate between different viewpoints, frequent use is made of the epipolar constraint, restricting the correspondence problem to a one-dimensional line search [SD96, ZKU+04]. Typically, either dense depth/disparity maps or complete 3D geometry models are estimated from the input image data in a pre-processing step. Zitnick et al. [ZKU+04] propose a view interpolation method based on depth maps. Seitz and Dyer [SD96] determine the F matrix to estimate dense disparity and warp-interpolate between two views of a static scene. An extension to dynamic scenes was proposed by Manning et al. [MD99], segmenting different motion layers by hand and restricting motions to rigid-body translations. Xiao et al. [XS04] describe an interpolation method for space and time between three images, based again on the assumption of rigidly moving objects. Most 3D geometry-based systems additionally rely on segmenting scene foreground for robust 3D geometry reconstruction [MBR+00, VBK05, SH07] or to make use of additional geometry models [DBY98, CTMS03, dAST+08]. Only the work of Vedula et al. is also capable of interpolating time. Having estimated 3D geometry, a number of different approaches have been proposed to suitably render novel views efficiently [LH96, MP04, LCL05, BBM+01, WWG06, WLW+05]. To correctly assess part of the contribution of our paper, it is important to realize that existing free-viewpoint navigation systems all rely on two conditions during acquisition: the cameras must be calibrated, and they must be synchronized. Else, epipolar geometry does not hold anymore in the time-varying case, and neither disparity/depth nor geometry can be recovered. Typically special camera hardware is necessary [WSLH02, YEBM02, WJV+05] for the acquisition.

Optical flow methods do not assume epipolar geometry constraints but assume color constancy between images. Thus, they are able to estimate the dense 2D motion field from two images alone [ZJK05, ST06, SC08, XCJ08].

Typically, the optical flow is used to compute the motion field between consecutive frames of an image sequence, i.e., in temporal direction. But the optical flow approach can also be applied to estimate the dense 2D correspondence field between images that are taken from different viewpoints. The advantage of dense correspondence field estimation is that no additional information is needed about relative camera positions or moments of exposure: both correspondence field estimation and subsequent image warping for rendering are purely image-based. Recently, a joint correspondence estimation and interpolation approach has been proposed that takes perceptual issues into account for improved visual results [SLW+08, SLAM08]. We adopted this method for our system. However, any other correspondence estimation method can be used as well.

Chapter 3

Acquisition

In our approach the acquisition of multiple videos is performed using unsynchronized and uncalibrated cameras, with the advantage that no special cameras and/or extra hardware are necessary. For the results presented in this paper we used up to 16 high resolution (1440 x 1080) HDV Canon XHA1 camcorders recording at 25 Hz mounted on standard tripods. The captured sequences are MPEG compressed and saved by the built in tape recorders and later transfered to a standard PC via firewire. This setup is very flexible, easy to setup, it runs completely on batteries and is thus suitable for out-door use.

The positioning of cameras is not constrained to any fix configuration and can be arranged in different forms according with the application (scene) needs. To make image-based interpolation between two acquired viewpoints possible, a sufficient amount of overlap between the two images is necessary. We achieve it by directing the cameras to determined region from the observed scene. The resulting spanning angle between two cameras varies between 10 and 15 degrees, in the vertical or horizontal direction. The total spanning angle depends on the configuration of the cameras.

Figure 3.1 shows three camera configurations that we used to film our videos. The first has 16 cameras arranged in 3 rows, 8 in the middle, 4 in the upper row and 4 in the lower row. The second arrangement is a 3×5 camera grid and the last one has 6 cameras arranged along an horizontal arc. A panoramic picture of the camera setup is taken from the center of the scene for later use, see Section 4.1.

Figure 3.1: Multiple camera setups for the acquisiton of spacetime video footage. Up to 16 off the shelf unsychronized HDV camcorders which recorded to tape on tripods where used. With this versatile setup indoor as well as outdoor recordings become feasible.

Chapter 4

Navigation Space

Image interpolation techniques such as optical flow estimation and perception-motivated image interpolation are very general because they can create transitions between two images without extra constraints (e.g. epipolar geometry or time synchronization). This allows to create view and/or time interpolation with unsynchronized and uncalibrated cameras. However, the movement of a virtual camera is then restricted by the transitions between the respective images. This is problematic in two ways:

First, movement is not independent. This means if the captured images are not spatially and temporarilay aligned, i.e. if unsynchronized and non-uniformly placed cameras are used, each motion between two images will be a composite of several degrees of freedom. Second, movement is not continuous. A global mapping is necessary that defines how more than two images can be interpolated such that a smooth navigation through the whole dataset is possible.

The main contribution of this work is the definition of an N-D space which allows independent and continuous navigation through multi-view video sequences recorded with uncalibrated and unsynchronized cameras. We will refer to this space as navigation space. One axis in this space represents for example, horizontal camera movement or the global scene time. A typical navigation space has two or three dimensions depending on the camera setup (cf Figure 4.2). Each image of the multi-view dataset is mapped to a point in this navigation space. The set of all images (from the different cameras) then forms a point cloud. The coordinates of an image in this space can be thought of as the viewing direction of the camera and the instant it was recorded measured in scene time. Our navigation space also bears some resemblence to the prevailing spacetime diagrams used by [Wol06] and [Inc08]. In the remainder of this section we introduce how these coordinates can be computed and how unrecorded points inside the convex hull spanned by the input can be described as a combination of recorded images.

4.1 Axis Definition

Our navigation space in general is not restricted to any specific camera arrangement and consists of view/space dimensions and a time dimension. Each image of the recorded sequences is mapped to a unique point in this space.

First, we search for a mapping of the camera/view positions to the navigation space. We denote this embedding the *camera manifold*. Depending on the camera setup an embedding that has one or two dimensions is usually sufficient. The criteria for this are that the manifold represents intuitive motion directions such as horizontal and vertical motion and that the distances between the cameras are preserved. Figure 4.1 depicts an example of the camera manifold that is used in the first of the camera arrangements shown in Figure 3.1. For the special case of stationary cameras, the embedding of the cameras is constant. Thus it can be easily obtained by a cylindrical mapping of the camera positions as recovered from a panorama image of the camera setup, Figure 3.1. For non-stationary cameras, the embedding is dynamic and changes in the time. In this case camera positions must be tracked over time and projected onto the manifold.

Figure 4.1: Typical camera manifold example. It corresponds with the 16 camera setup in Fig. 3.1. Each camera is represented by a cube on the xy-plane.

For dynamic scenes one axis of the navigation space is the scene time axis. The time coordinate for each image of a camera is determined by the local camera time of each image and the mapping of the camera time to the global scene time. In contrast to enforcing time synchronicity of the recording cameras, a one to one correspondence between frames of different cameras is not required. Instead it suffices that the time relation is known (see also Section 4.3). Thus, even with unsynchronized cameras time independent changes, such as time-freeze view changes, are possible.

After the mapping, all recorded images from the multi-video sequence form a point cloud in navigation space, Figure 4.2. The convex hull of this point cloud bounds the space of positions, and thus virtual cameras, that can be rendered with our approach.

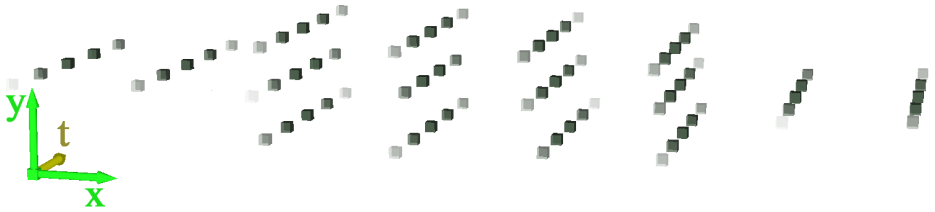

Figure 4.2: The still images from the multi-video sequence form a point cloud in navigation space. Images captured by the same camera are easily identified, as they lie on a straight line that runs across the temporal axis.

4.2 Tetrahedralization

To smoothly move through navigation space, we define a virtual camera that is controlled interactively by the user. The position of the camera is represented by a point in the navigation space. Note, that since view and time dimensions are handled equally, this already includes the interpolation in both view and time.

Following a pure image-based approach, the virtual camera must be described as a weighted combination of recorded images from real cameras. To accomplish this, first a delaunay tetrahedralization of the multi-video point cloud is computed (cf. Figure 4.3). [1] This ensures that each spacetime point is inside at least one *spacetime tetrahedron*. For each tetrahedron T, the vertices v_1, v_1, v_3, v_4 , refered to as *images vertices*, represent recorded frames from the multi-view sequence. The edges define adjacency between the vertices images. During rendering the dense correspondence fields defined by these edges are used to warp the images.

Figure 4.3: Spacetime tetrahedra. The delaunay tetrahedralization ensures that each point inside the convex hull of the point cloud lies in one tetrahedron. Edges between vertices images define adjacency between them

The problem of rendering a novel spacetime point for a given virtual

[1] For the sake of simplicity we refer to the more general case of tetrahedra, the simplex in three dimensional space. However, please note that our approach can be also be directly applied to two or even higher dimensional navigation spaces.

camera is now two part. First we search for a tetrahedron that contains it. Then, the virtual camera is described in terms of vertices images, adjacency between the vertices images and barycentric coordinates. Section 5 explains in detail how the image is finally rendered with multi-image interpolation.

4.3 Time Shifts

With synchronized cameras, all acquired videos begin at the same moment and a one to one correspondence between frames recorded with different cameras and the global scene time exists. For unsynchronized cameras this is not the case. However, as discussed in Section 4.1, we can still map local camera time to global scene time if the time offset of the camera c, δ_t^c is known and all cameras record at the same frame rate. For fast dynamic scenes, not only frame correspondence between the sequences must be established, but subframe accuracy is also needed.

Different approaches have been proposed to determine the time offset between the recorded videos to subframe accuracy. For our experiments we used two methods, depending on the recording setup and scene. The first method, presented in [MSMP08], estimates sub-frame temporal offsets between unsynchronized, non-stationary cameras using motion trajectory correspondences. This method works sufficiently accurate in general but has some restrictions: First, a scene point with sufficient motion must be tracked over a longer period in order to estimate the frame-accurate time offset. Second, the subframe-accurate time offset estimation is dependent on the quality of estimated fundamental matrices which can become unstable.

Another solution is to use the audio capabilities available with camcorders. While audio is recorded in sync with the video data, it has a much higher temporal sampling rate, 48 kHz compared to 25 Hz in our example. Thus, measuring the recording time of known time-audio markers results in a very robust and subframe-accurate time offset estimation. Some limitations for this method include open air acquisition where cameras are placed far from each other and the traveling time of sound becomes noticeable . Also, configurations where the sound cannot propagate uniformly to the cameras or when recording an audio signal in sync with the video signal make its application impossible. We used both methods to estimate the time offsets of our unsynchronized camera setups, depending of arrangement. Section 6 gives an overview of our experiments and which synchronization method was used.

Once the time offset δ_t^c for each camera has been estimated, the local camera time is mapped to the global scene time t_g as follows:

$$t_g = t_c + \delta_t^c \tag{4.1}$$

This mapping makes it possible to navigate through unsynchronized multi-

view sequences in the same way as sequences from synchronized cameras. Figure 4.4 shows the navigation space with and without estimated time shifts of the cameras, respectively.

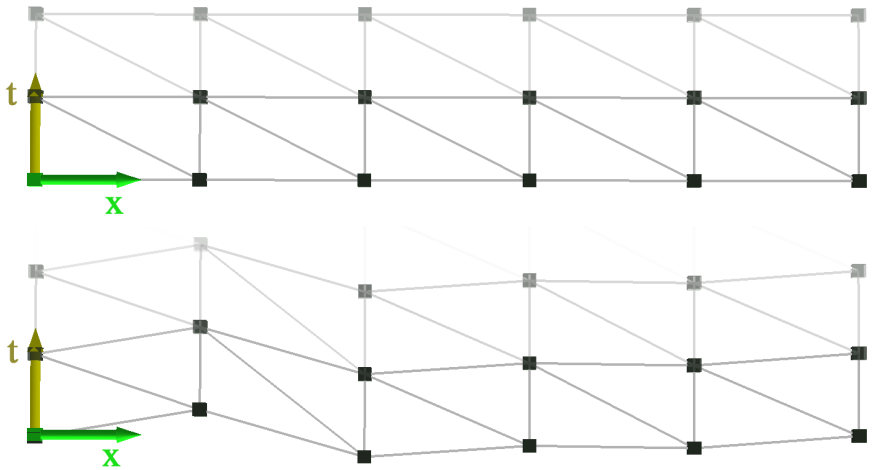

Figure 4.4: Navigation space with and without estimated time shifts of the cameras. This correction makes it possible to navigate through unsynchronized multi-view sequences in the same way as sequences recorded with synchronized cameras.

4.4 Correspondence Field Estimation

Our rendering as described in Section 5 is implemented as multi-image interpolation. It is therefore based on estimated dense correspondence fields between adjacent vertices images as defined by the tetrahedralization of the vertices images. The set of all edges thus represents all dense correspondences between all pairs of images that must be estimated during preprocessing. Several methods to compute dense correspondences between pairs of images can serve as the input to our method. Since we are dealing with interpolations of real-world scenes, it is mandatory that the method also computes warp fields in the presence of occlusion. In this paper we computed the warps with a recent perception motivated method described in [SLW+08, SLAM08]. The advantage of this method is that it is very fast and robust, while also allowing for the manual correction of errors (cf. Section 4.5). However, other methods such as optical flow techniques that handle occlusions such as [ZJK05, ST06, SC08, XCJ08], could also be used.

4.5 Quality Improvements

Manual Correction of Correspondence Fields. Automatic methods to estimate dense correspondence fields can deliver unsatisfactory results between specific frames, especially when large parts get occluded. However, in these specific cases it is often feasible to manually correct errors in the order of minutes. In this way one can achieve a significant improvment of the final results at very low cost.

In addition to the possiblities, that some automatic estimation approaches allow, we have implemented a tool which assists in manual corrections of the estimated correspondence fields. The idea hereby is to edit the resulting vector field directly by manually selecting regions of a correspondence and assuming translation between these. Additionally, for dynamic scenes with static background we can use background-foreground segmentation to replace the background correspondences by previously corrected correspondences.

Color Correction, despite using the same camera model for recording, the individual input videos often vary significantly in color. In order to reduce this variance, we apply a simple color correction similar to the method proposed by [SGSS08]. It is based on a gain and offset model where the parameters are defined relative to a reference camera. The corrections are applied on the fly during rendering.

Optimal model parameters per color channel are estimated in a least-squares sense from the already estimated dense correspondences. To further increase the robustness of the estimation we filter outliers with the RANSAC algorithm [FB81] but divide the images in a 4×4 grid for sample selection. This avoids concentrating the estimation on large uniform color regions. Figure 4.5 shows an example of images acquired from two different cameras, before and after the color correction.

Spatial Alignment, even if the cameras are pointed towards the same point in the scene, small misalignments due to imprecision are visible. We correct these misalignments by panning the images from each camera, so that a user defined point coincides. These per-camera panning offsets are applied to all frames and correspondence fields of the data set.

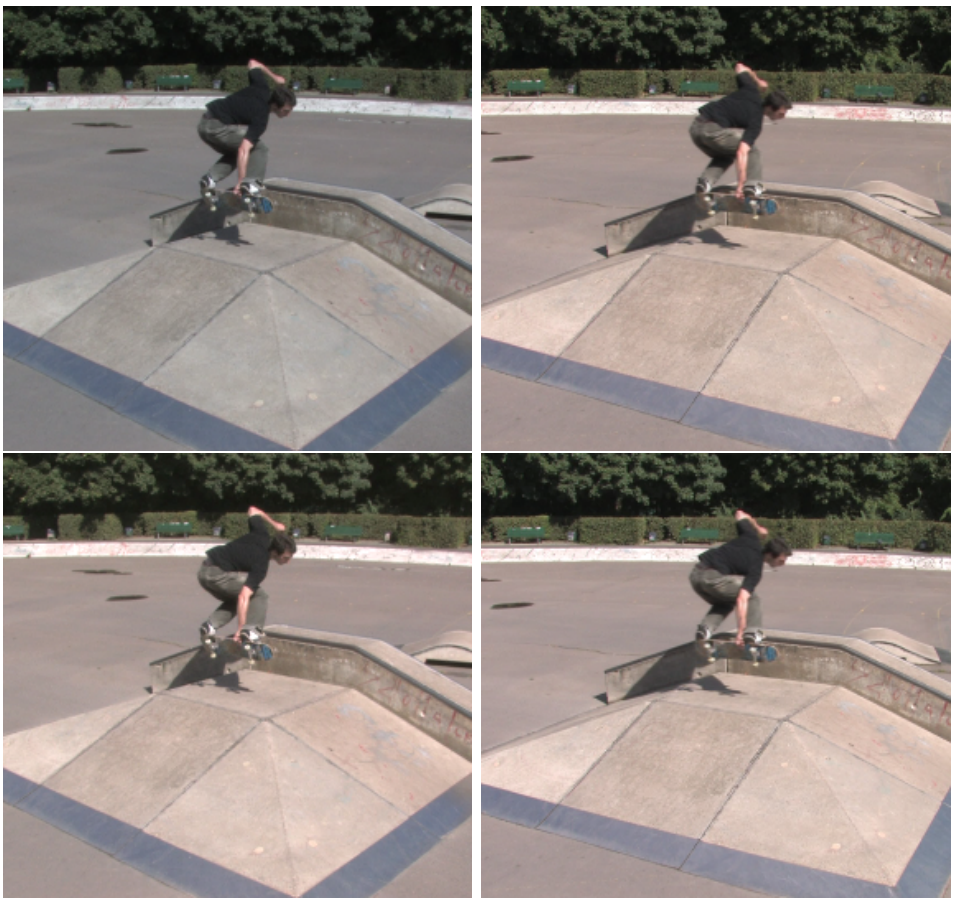

Figure 4.5: The first two images shows camera c_1 and c_2, before the color balance and the last two images show c_1 and c_2 after the color balance.

Chapter 5

Rendering

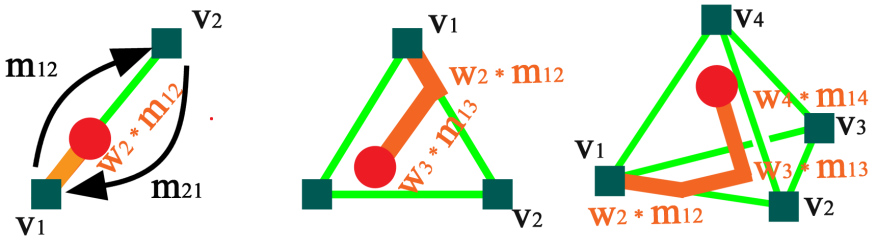

Figure 5.1: Correspondence maps m_{ij} are used to render novel images between two points v_i and v_j (left). We extend this to an N-dimensional space where a weighted combination of correspondence maps can be used to render any point inside a triangle (middle), tetrahedron (right) or any simplex of a higher dimension

After pre-processing, we are able to render any intermediate view and/or time point in real-time using multi-image interpolation. While in the following we concentrate on the 3D navigation space case, our approach works equally for all dimensions.

Each point inside the navigation space lies in a simplex, in the $N = 3$ case a tetrahedron T, after delaunay tetrahedralization. In Section 4 we have described how the virtual camera is represented as a point p in navigation space. The first step to render q, the image represented by point p in navigation space, is to search for a tetrahedron T that contains p. Then the four $(N + 1)$ images-vertices $v_i, i = 1, ..., 4$ and the four $((n - 1)n/2)$ edges of T define a set of images and dense correspondence fields m_{ij} and m_{ji}. Note that all m_{i*} are defined in the same image basis, and can thus be combined. Then q can be computed by multi-image interpolation as follows: First the vertices images are warped as

$$q_i = v_i \circ (w_j m_{ij} + w_k m_{ik} + w_l m_{il}) \tag{5.1}$$

where $i, j, k, l = 1, ..., 4$, $i \neq j$, $i \neq k$, $i \neq l$, $j \neq k$, $j \neq l$, $k \neq l$ and the w_* are the barycentric coordinates of p in relation to T (cf. Figure 5.1). The operator $\circ$ of an image and a (weighted) correspondence field is the forward warping of each pixel as defined by the correspondence field. Then, the four (N+1) vertices images are blended according to their barycentric coordinates.

$$q = \sum_{i=1}^{i=N+1} w_i * q_i \tag{5.2}$$

Despite a point may lie in more than one tetrahedron at the same time, i.e. if the point p lies exactly on an edge or on one of the corner points we stop our search when we find the first tetrahedron that contains p. In this case, one or more barycentric coordinates are zero and there is no difference, whatever the tetrahedron we use to define p.

Additionally, we handle disoccluded parts during rendering similar to [SLAM08]. Based on the divergence of the dense correspondence field we locally adjust the blending weights such that missing parts in one images have no influence on the final blending result. Figure 5.2 shows an example of our rendering approach. As a last step, the borders of the rendered images are cropped, since no reliable correspondence information is available for these regions.

Our rendering algorithm was implemented on the GPU. At 940×560 we achieved 35 fps on an NVIDIA GeForce 8800 GTX.

5.1 Occlusion Handling

Due to occlusion during interpolation more than one pixel can be warped to the same location. Then the foreground pixel has to be rendered and the background pixel must be discarded. With a pure image-based approach however no depth information of the pixels is available, and thus other criteria must be used to resolve this issue. Typically, the fastest pixel is assumed to be in the foreground which is often a good heuristic. Since we have more information about the camera setup and the images avaiable, we can additionally make use of this information. For a camera and its right neighbor, for example, the X-component of the correspondence map between images of these cameras can be assumed to represent a large portion of the induced disparity. Thus we use it readily to resolve these conflicts. The same reasoning holds for vertical neighbors and the Y-component.

5.2 Navigation Interface

Our navigation interface has two main components, a spacetime tetrahedra viewer and an image space viewer. As mentioned before, each point inside

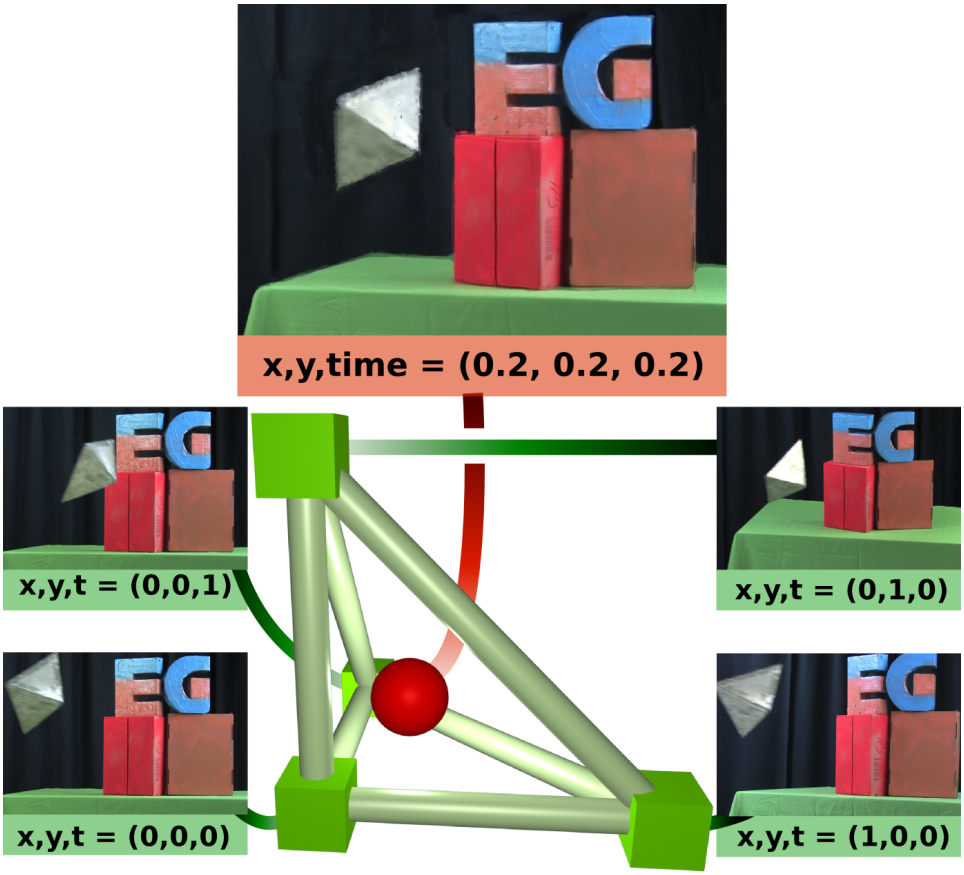

Figure 5.2: Example of view and time interpolation. Four images v_1, v_2, v_3, v_4 (bottom), represented by cubes, are warped and blended into the output image q (top) which is defined by the red ball p in navigation space.

the navigation space is mapped to an image in the image space. Therefore, our system allows two interconnected possibilities of navigation: One direct in image space and other using the navigation space.

The first navigation mode is a very intuitive navigation in the image space viewer. By clicking and dragging with the left mouse button, one can fly along the cameras views.Using the middle button, it is possible to scroll through the time. The navigation in all directions (space and time) is constrained only by the convex hull of the multi-video data in the navigation space. Along with this submission, our interactive spacetime viewer software is made available. It demonstrates the real-time navigation possibilities described above.

Another alternative to navigate through the spacetime is using directly the spacetime tetrahedra viewer. It can be done by placing keyframes. The

keyframes are defined by a virtual camera position in the navigation space and the frame number in the output video. An interpolation mode defines how the in-between frames are computed, if using linear interpolation or Catmull-Rom splines (Fig. 5.3). The spacetime trajectories can mix linear inrepolation and splines in the same trajectory. Finally, videos can be rendered by following the view and time coordinates described in the spacetime trajectory.

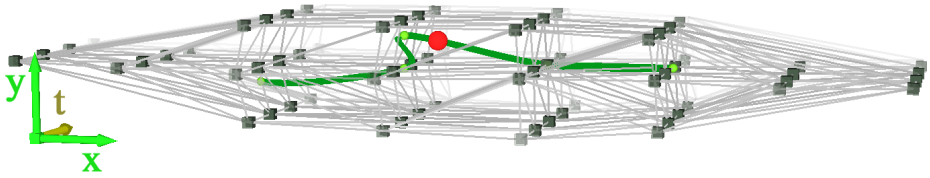

Figure 5.3: Navigation using trajectories in the spacetime tetrahedra viewer. The trajectories are defined by keyframes inside the convex hull of the multi-video data.

Chapter 6

Results

To test our system, we have recorded five different real-world scenes, Fig. 6.1. We have selected these scenes to evaluate system performance for different challenging situations. Unfortunately, still images are not able to convey meaningful information about the quality of our results.

Among our different test scenes, the *dancer* is the most conventional recording. She was filmed using sixteen cameras, eight in the middle row and four each in an upper and lower row, Fig. 3.1. Cameras are spaced roughly 10 degrees apart, approximately centered on the stage, recording with wide-angle lenses to capture the entire scene. Time offsets were estimated using the method presented by Meyer et al. [MSMP08].

The *glass of water* scene is more demanding as it features strong refraction and reflection effects. It was recorded using five cameras arranged along a horizontal line spaced about 10° apart. The cameras are roughly centered on the spilling water, all zoomed in to record fine details. Time offsets were determined using an audio signal, cf. Sect.4.3.

We recorded a *fire-breather* to evaluate performance in the presence of high dynamic contrast (actually, over-exposure) and volumetric phenomena without a well-defined surface. Camera arrangement was the same as for the *dancer* scene, as was the method for determining time offsets.

The *beer can* scene is the most demanding due to the high speed of the spraying foam as well as the multitude of tiny droplets forming a semi-transparent mist that would defy any geometric modeling approach. The scene was acquired using a grid arrangement of 3×5 cameras, all roughly centered on the can and covering a solid angle of approximately $30° \times 50°$. Time offsets were estimated using an audio signal.

Our *skateboarder* scene was recorded one sunny afternoon in a park nearby. We carried six video cameras plus tripods there, set them up in a horizontal arc of approximately 60° centered on the skater ramp, pressed 'record' on all cameras, gave an audio signal for synchronization, sat down in the sun and watched the skaters do their magic.

To accelerate correspondence field estimation, we downsampled all images to 480×270 pixels prior to pre-processing. We found that the resulting lower-resolution correspondence maps nevertheless suffice to render convincing results at full HD resolution.

6.1 Limitations

In comparison to free-viewpoint navigation systems based on 3D geometry [CTMS03, VBK05], it is clear that our system is restricted to viewpoints that lie somewhere in-between camera positions. We can only navigate inside the convex hull spanned by all camera positions, looking at the scene from different directions, but we cannot, for example, fly through the scene.

The output rendering quality is determined by the accuracy of the estimated correspondence fields and the interpolation model. Since a truly reliable, automatic method for dense correspondence estimation has yet to be found, we provide for human intervention to correct for correspondence errors, Sect. 4.5. With respect to the interpolation model we currently employ, we are limited to scenes that feature only one motion direction per pixel: the current interpolation model does not provide for, e.g., semi-transparent objects behind each other that move in different directions.

To obtain good pre-processing results, we observed that the angle between adjacent cameras should not exceed $\approx 15°$. Above $15°$, image differences seem to become too large for convincing interpolation results. The same is true for too fast scene motion. As a rule of thumb, we found that correspondences over either space or time should not be farther apart than approximately 10% of linear image size.

Figure 6.1: One video frame from each of our test sequences. Besides one rather conventional scene (*dancer*), we have deliberately chosen complex scenes that pose a variety of challenges: outdoor capture (*skateboarder*), volumetric effect and high dynamic contrast (*fire-breather*), fast, non-rigid motion (*beer can*), and reflection and refraction (*glass of water*). To assess the quality of our system's results, we refer the reader to the accompanying video.

Chapter 7

Conclusion

We have presented a purely image-based rendering system to viewpoint-navigate through space and time of arbitrary dynamic scenes. Our system works with unsynchronized and uncalibrated multi-video data and requires only a modest number of cameras. Since we do not need special acquisition hardware or scene-invasive setup procedures, we can record access-restricted scenes, e.g. professional sports events, in uncontrolled environments, e.g. outdoors. Our approach allows for smooth interpolation of view perspective and time, i.e., for simultaneous free-viewpoint and slow motion rendering. Finally, we have demonstrated that our system can handle complex scenes for which depth or geometry might not be well-defined.

As next steps, we want to extend our system to facilitate working with input data from non-stationary cameras, e.g., mobile phone cameras. This will probably involve some kind of camera tracking in order to be able to adapt navigation space tetrahedralization correspondingly. Along a different research direction, our system lends itself to creating a wide range of special effects. Besides time-freeze and slow motion rendering already presented in this paper, we plan to build an F/X editor that enables creating advanced effects such as motion streaks, space blur, or space-time ramps.

Bibliography

[BBM+01] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured Lumigraph Rendering. In *Proc. ACM Conference on Computer Graphics (SIGGRAPH'01),* Los Angeles, pages 425–432. ACM, 2001.

[CTMS03] J. Carranza, C. Theobalt, M. Magnor, and H. P. Seidel. Free-Viewpoint Video of Human Actors. In *Proc. ACM Conference on Computer Graphics (SIGGRAPH'03),* San Diego, pages 569–577. ACM, 2003.

[dAST+08] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance Capture from Sparse Multi-View Video. *ACM Trans. Graph.*, 27(3):1–10, 2008.

[DBY98] P. Debevec, G. Borshukov, and Y. Yu. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. In *Proceedings Eurographics Rendering Workshop (EGRW)*, pages 105–116, 1998.

[FB81] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[Inc08] Digital Air Inc. Digital Air Techniques. http://www.digitalair.com/techniques/index.html, 2008.

[LCL05] Jian-Guang Lou, Hua Cai, and Jiang Li. A Real-Time Interactive Multi-View Video System. In *Proceedings of the 13th annual ACM international conference on Multimedia (MULTIMEDIA)*, pages 161–170, New York, NY, USA, 2005. ACM.

[LH96] M. Levoy and P. Hanrahan. Light Field Rendering. In *Proceedings ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 31–42, 1996.

[MBR+00] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-Based Visual Hulls. In *Proceedings ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 369–374, 2000.

[MD99] R.A. Manning and C.R. Dyer. Interpolating View and Scene Motion by Dynamic View Morphing. *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1:–394 Vol. 1, 1999.

[MP04] W. Matusik and H. Pfister. 3D TV: A Scalable System for Real-Time Acquisition, Transmission, and Autostereoscopic Display of Dynamic Scenes. In *Proceedings ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 814–824, 2004.

[MSMP08] Benjamin Meyer, Timo Stich, Marcus Magnor, and Marc Pollefeys. Subframe Temporal Alignment of Non-Stationary Cameras. In *Proc. British Machine Vision Conference BMVC '08*, 2008.

[SC08] T. Schoenemann and D. Cremers. High Resolution Motion Layer Decomposition using Dual-space Graph Cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, June 2008.

[SD96] Steven M. Seitz and Charles R. Dyer. View Morphing. In *Proc. SIGGRAPH 96*, pages 21–30, 1996.

[SGSS08] Noah Snavely, Rahul Garg, Steven M. Seitz, and Richard Szeliski. Finding Paths through the World's Photos. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, 27(3):11–21, 2008.

[SH07] J. Starck and A. Hilton. Surface Capture for Performance Based Animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.

[SLAM08] Timo Stich, Christian Linz, Georgia Albuquerque, and Marcus Magnor. View and Time Interpolation in Image Space. *Computer Graphics Forum (Proc. Pacific Graphics)*, 27(7), 2008.

[SLW+08] Timo Stich, Christian Linz, Christian Wallraven, Douglas Cunningham, and Marcus Magnor. Perception-motivated Interpolation of Image Sequences. In *Proc. ACM Symposium on Applied Perception in Graphics and Visualization (APGV)*, pages 97–106, Los Angeles, USA, 2008. ACM.

[ST06] Peter Sand and Seth Teller. Particle Video: Long-Range Motion Estimation using Point Trajectories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2195–2202, Washington, DC, USA, 2006. IEEE Computer Society.

[VBK05] S. Vedula, S. Baker, and T. Kanade. Image Based Spatio-Temporal Modeling and View Interpolation of Dynamic Events. *ACM Transactions on Graphics*, 24(2):240–261, April 2005.

[WJV+05] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High Performance Imaging using large Camera Arrays. *ACM Trans. Graph.*, 24(3):765–776, 2005.

[WLW+05] S. Wrmlin, E. Lamboray, M. Waschbsch, P. Kaufmann, A. Smolic, and M. Gross. Image-Space Free-Viewpoint Video. In *Proc. of Vision, Modeling, Visualization (VMV)*, pages 453–460, 2005.

[Wol06] M. Wolf. Space, Time, Frame, Cinema: Exploring the Possibilities of Spatiotemporal Effects. *New Review of Film and Television Studies*, pages 369–374, December 2006. www.digitalair.com/techniques/STFC.pdf.

[WSLH02] Bennett Wilburn, Michal Smulski, Kelin Lee, and Mark A. Horowitz. The Light Field Video Camera. In *SPIE Media Processors 2002*, pages 29–36, 2002.

[WWG06] M. Waschbsch, S. Wrmlin, and M. Gross. Interactive 3D Video Editing. *The Visual Computer 22, 9-11 (Proceedings of Pacific Graphics 2006)*, pages 631–641, 2006.

[XCJ08] Li Xu, Jianing Chen, and Jiaya Jia. A Segmentation Based Variational Model for Accurate Optical Flow Estimation. In *Proc. ECCV 2008*, 2008.

[XS04] Jiangjian Xiao and Mubarak Shah. Tri-View Morphing. *Computer Vision and Image Understanding*, 96:345–366, 2004.

[YEBM02] Jason C. Yang, Matthew Everett, Chris Buehler, and Leonard Mcmillan. A Real-Time distributed Light Field Camera. In *Eurographics Association*, pages 77–86, 2002.

[ZJK05] C.W. Zitnick, N. Jojic, and Sing Bing Kang. Consistent Segmentation for Optical Flow Estimation. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2:1308–1315 Vol. 2, Oct. 2005.

[ZKU+04] C. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-Quality Video View Interpolation Using a Layered Representation. In *Proc. ACM Conference on Computer Graphics (SIGGRAPH'04),* Los Angeles, pages 600–608. ACM, 2004.